Package 'diagFDR'

April 27, 2026

Title Verifiable FDR Diagnostics for Proteomics

Version 0.1.1

Description Provides methods to compute verifiable false discovery rate (FDR) diagnostic checks for workflows based on target-decoy competition and related confidence measures. Implements calibration, stability and tail diagnostics, including tail support, threshold elasticity, posterior error probability (PEP) reliability, and equal-chance checks. If you used this package in your research, please cite the associated preprint <doi:10.64898/2026.04.16.718468>. Detailed examples of using this package can also be found on the GitHub repository (<https://github.com/Jacky11/diagFDR>).

URL https://www.biorxiv.org/content/10.64898/2026.04.16.718468

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.2.0)

Imports dplyr, tibble, ggplot2, purrr, rlang, readr, xml2, cp4p, tidyr, scales, data.table

Suggests arrow, knitr, officer, rvg, testthat (>= 3.0.0), withr, rmarkdown, jsonlite

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Quentin Giai Gianetto [aut, cre]

Maintainer Quentin Giai Gianetto <quentin.giaigianetto@pasteur.fr>

Repository CRAN

Date/Publication 2026-04-27 18:30:08 UTC

Contents

`as_dfdr_tbl` *Create a* `dfdr_tbl`

Description

Coerces input to a tibble, standardizes the `is_decoy` column to logical, validates required columns and types, attaches metadata, and returns an S3 object of class `dfdr_tbl` for downstream diagnostics.

Usage

```
as_dfdr_tbl(
  x,
  unit = NA_character_,
  scope = NA_character_,
  q_source = NA_character_,
  q_max_export = NA_real_,
  p_source = NA_character_,
  provenance = list()
)
```

Arguments

`x`	A data.frame with columns `id`, `is_decoy`, `q`, `pep`, `run`, `score` (where `pep`/`run`/`score` may be `NA` if unavailable). Optionally may contain `p`.
`unit`	Character. Statistical unit, e.g. `"psm"`, `"precursor"`, `"peptide"`, `"protein"`.
`scope`	Character. Scope of FDR control, e.g. `"runwise"`, `"global"`, `"aggregated"`.
`q_source`	Character. Description of where q-values came from (e.g. a column name or tool).
`q_max_export`	Numeric. Optional export ceiling used to generate q (if known).
`p_source`	Optional character describing where `p` came from.
`provenance`	Named list carrying provenance information (tool, version, parameters, command, etc.).

Value

An S3 object of class `dfdr_tbl` that inherits from tbl_df (and `data.frame`). The returned object:

- contains the validated input data with `is_decoy` coerced to logical (`TRUE`/`FALSE`);
- carries metadata in `attr(x, "meta")` with entries `unit`, `scope`, `q_source`, `q_max_export`, `p_source`, and `provenance`.

Examples

```
library(tibble)

df <- tibble(
  id = as.character(1:6),
  run = c("run1","run1","run1","run2","run2","run2"),
  is_decoy = c(FALSE, FALSE, TRUE, FALSE, TRUE, FALSE),
  score = c(10, 9, 8, 7, 6, 5),
  q = c(0.01, 0.02, 0.02, 0.05, 0.06, 0.07),
  pep = c(0.01, 0.02, NA, 0.05, NA, 0.07)
)

x <- as_dfdr_tbl(
  df,
  unit = "psm",
  scope = "global",
  q_source = "toy_q",
  provenance = list(tool = "toy")
)

x
attr(x, "meta")
```

---

dfdr_bh_diagnostics *BH diagnostics at a headline alpha (threshold, discoveries, boundary support)*

Description

Computes the Benjamini–Hochberg (BH) rejection threshold t_α, the number of discoveries R_α, and a simple "boundary support" measure: the number of p-values just above t_α. Boundary support is intended to indicate how sensitive the discovery set may be to small perturbations of p-values near the cutoff.

Usage

```
dfdr_bh_diagnostics(
  x,
  alpha = 0.01,
  boundary = c("mult", "add"),
  win = 0.2,
  delta = NULL
)
```

Arguments

x	A `dfdr_tbl` (or data frame) containing columns `id` and `p`.

alpha	Numeric BH FDR level in $(0, 1)$.
boundary	Character. One of "mult" (multiplicative window) or "add" (additive window).
win	Numeric. Relative width for the multiplicative boundary window (default 0.2). Used when boundary = "mult", defining the interval $(t_\alpha, (1 + \text{win})t_\alpha]$.
delta	Numeric. Additive width for the additive boundary window. Required when boundary = "add", defining the interval $(t_\alpha, t_\alpha + \delta]$.

Value

A list with two elements:

headline A one-row tibble containing BH summary diagnostics, including t_alpha (the BH threshold), R_alpha (the number of rejected hypotheses / discoveries), and N_boundary (the number of p-values in a right-neighborhood above the cutoff as determined by boundary, win, and delta).

accepted A character vector of id values corresponding to discoveries (rows with p <= t_alpha). If no finite BH threshold exists, this is character(0).

Examples

```
library(tibble)

set.seed(1)
n <- 5000
x <- tibble(
  id = as.character(seq_len(n)),
  p = c(stats::runif(4500), stats::rbeta(500, 0.3, 1))
)

out <- dfdr_bh_diagnostics(x, alpha = 0.01, boundary = "mult", win = 0.2)
out$headline
length(out$accepted)
```

---

dfdr_bh_elasticity *BH list elasticity (Jaccard) across alpha values*

Description

Computes the stability (elasticity) of Benjamini–Hochberg (BH) discovery sets under a multiplicative perturbation of the nominal FDR level. For each alpha in alphas, the function compares the BH discovery set at alpha to the discovery set at (1+eps)*alpha using the Jaccard index.

Usage

```
dfdr_bh_elasticity(
  x,
  alphas = c(1e-04, 5e-04, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05),
  eps = 0.2
)
```

Arguments

x	A dfdr_tbl (or data frame) with columns id and p. Only finite p-values are used.
alphas	Numeric vector of BH levels in $(0, 1)$.
eps	Numeric scalar $\epsilon > 0$. Relative perturbation (default 0.2), i.e. compares alpha vs (1+eps)*alpha.

Value

A tibble with one row per element of alphas. Columns include:

alpha The nominal BH level.

alpha_perturbed The perturbed level min((1+eps)*alpha, 1).

t_alpha BH rejection threshold at alpha (NA if no rejections).

t_alpha_perturbed BH rejection threshold at alpha_perturbed (NA if none).

R_alpha Number of discoveries at alpha.

R_alpha_perturbed Number of discoveries at alpha_perturbed.

jaccard Jaccard similarity between the two discovery ID sets.

This output is intended to quantify how sensitive BH discoveries are to small changes in the chosen FDR level.

Examples

```
library(tibble)

set.seed(1)
n <- 5000
x <- tibble(
  id = as.character(seq_len(n)),
  # mixture: mostly null p-values + a small enriched component
  p = c(stats::runif(4500), stats::rbeta(500, 0.3, 1))
)

dfdr_bh_elasticity(x, alphas = c(1e-3, 5e-3, 1e-2, 2e-2), eps = 0.2)
```

---

dfdr_curve_stability *Stability curve across alpha values*

Description

Evaluates dfdr_headline over a grid of FDR thresholds (alphas) to obtain a stability curve that can be plotted or compared across workflows.

Usage

```
dfdr_curve_stability(x, alphas, k_fixed = 10, k_sqrt_mult = 2)
```

Arguments

x	An dfdr_tbl.
alphas	Numeric vector of FDR thresholds in $(0, 1]$.
k_fixed	Integer. Fixed +/-K sensitivity interval used by dfdr_headline.
k_sqrt_mult	Numeric. Multiplier for the adaptive +/- ceiling(mult*sqrt(D)) interval used by dfdr_headline.

Value

A tibble with one row per value of alphas. Columns are those returned by dfdr_headline plus the corresponding alpha. The table summarizes how headline quantities (e.g., accepted target/decoy counts and sensitivity diagnostics) vary with the chosen FDR cutoff.

Examples

```
library(tibble)

set.seed(1)
n <- 5000
df <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  pep = NA_real_
)

df$q <- diagFDR:::tdc_qvalues(df$score, df$is_decoy, add_decoy = 1L)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy_tdc")

dfdr_curve_stability(x, alphas = c(1e-3, 5e-3, 1e-2))
```

---

dfdr_elasticity *Threshold elasticity (list stability to cutoff perturbation)*

Description

Computes the stability of the accepted target set under a multiplicative perturbation of the threshold:

$$S_\alpha(\epsilon) = J(A(\alpha), A((1+\epsilon)\alpha)),$$

where J is the Jaccard index and $A(\alpha)$ is the set of accepted target IDs at threshold α.

Usage

```
dfdr_elasticity(x, alphas, eps = 0.2)
```

Arguments

x	An dfdr_tbl.
alphas	Numeric vector of FDR thresholds in $(0, 1]$.
eps	Numeric scalar $\epsilon \geq 0$. Relative perturbation (e.g. 0.2 compares α vs 1.2α).

Value

A tibble with one row per alpha. Columns include:

alpha The baseline threshold.

eps The relative perturbation.

n1 Number of accepted targets at alpha.

n2 Number of accepted targets at (1+eps)*alpha.

jaccard Jaccard overlap between the two accepted target sets.

Examples

```
library(tibble)

set.seed(1)
n <- 5000
df <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  pep = NA_real_
)

df$q <- diagFDR:::tdc_qvalues(df$score, df$is_decoy, add_decoy = 1L)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy_tdc")

dfdr_elasticity(x, alphas = c(1e-3, 5e-3, 1e-2), eps = 0.2)
```

---

dfdr_equal_chance_qbands

Equal-chance plausibility by q-value bands

Description

Computes decoy fractions in q-value bands and performs a pooled binomial test of whether the decoy fraction is near 0.5 in a mismatch-dominated region (specified by low_conf).

Usage

```
dfdr_equal_chance_qbands(
  x,
  breaks = c(0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5),
  low_conf = c(0.2, 0.5),
  min_N = 2000
)
```

Arguments

x	An dfdr_tbl.
breaks	Numeric vector of q-value cutpoints used to form bands.
low_conf	Length-2 numeric vector giving the pooled test interval (e.g. c(0.2, 0.5)).
min_N	Integer. Minimum pooled sample size required for the pooled test to be considered stable.

Value

A list with components:

bands A tibble with one row per q-band, including n (band size), n_decoy, decoy_frac, and q_mean.

pooled A one-row tibble with pooled quantities over bands whose q_mean lies within low_conf, including pi_D_hat (pooled decoy fraction), a Wilson confidence interval, and a binomial test p-value. pass_minN indicates whether min_N is met. In cases where the requested banding is not applicable (e.g. qmax below the smallest nonzero break), fields are returned as NA and a note may be provided.

params A list of parameters used (breaks, low_conf, min_N).

Examples

```
library(tibble)

set.seed(1)
n <- 8000
df <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.9, 0.1)),
  score = rnorm(n),
  pep = NA_real_
)
# Toy q-values in [0, 1]
df$q <- stats::runif(n)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy")

ec <- dfdr_equal_chance_qbands(x, breaks = c(0, 0.2, 0.5, 1), low_conf = c(0.2, 0.5), min_N = 200)
ec$pooled
```

dfdr_headline *Headline stability diagnostics at a given alpha*

Description

Computes headline counts at the acceptance threshold corresponding to `alpha` (e.g., numbers of accepted targets/decoys) and simple stability diagnostics that quantify how sensitive the result is to small changes in the decoy boundary support.

Usage

```
dfdr_headline(x, alpha = 0.01, k_fixed = 10, k_sqrt_mult = 2)
```

Arguments

`x`	A `dfdr_tbl` containing at least `id`, `is_decoy`, and a q-value column (see Details).
`alpha`	Numeric FDR threshold in $(0, 1]$ at which headline metrics are computed.
`k_fixed`	Integer. Fixed perturbation size used for sensitivity analyses based on $D \pm k$ around the decoy count at the threshold.
`k_sqrt_mult`	Numeric. Multiplier for adaptive perturbations of size $\lceil \text{mult}\sqrt{D} \rceil$.

Details

The function assumes that `x` contains target/decoy labels in `is_decoy` and q-values in column `q`. If your input uses another name (e.g., `q_value`), rename it to `q` before calling this function.

Value

A one-row tibble with headline diagnostics evaluated at the specified `alpha`. The table includes the number of accepted targets and decoys at the threshold and additional sensitivity/stability summaries derived from perturbing the boundary decoy support (using `k_fixed` and `k_sqrt_mult`). The returned tibble is intended for reporting and for comparison across workflows.

Examples

```
library(tibble)

set.seed(1)
n <- 5000
x <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  pep = NA_real_
)
```

```
# Construct simple TDC q-values from the score (higher is better)
x$q <- diagFDR:::tdc_qvalues(score = x$score, is_decoy = x$is_decoy, add_decoy = 1L)
x <- as_dfdr_tbl(x, unit = "psm", scope = "global", q_source = "toy_tdc")

dfdr_headline(x, alpha = 0.01)
```

`dfdr_local_tail` *Local tail support near the cutoff*

Description

Counts the number of observations (and decoys) in a right-neighborhood of each threshold α, defined as $(\alpha, \alpha + \rho\alpha]$ where ρ is `win_rel`. This approximates how well supported the decision boundary is by decoys in the immediate tail.

Usage

```
dfdr_local_tail(
  x,
  alphas,
  win_rel = 0.2,
  truncation = c("warn_drop", "drop", "cap")
)
```

Arguments

`x`	An `dfdr_tbl`.
`alphas`	Numeric vector of FDR thresholds in $(0, 1]$.
`win_rel`	Numeric. Relative window width ρ (default 0.2).
`truncation`	Character. How to handle cases where $\alpha + \rho\alpha$ exceeds the available q-value export range (i.e. `q_max_export`). One of `"warn_drop"`, `"drop"`, `"cap"`.

Value

A tibble with one row per `alpha`. Columns include:

alpha The threshold.

delta The effective window width used ($\rho\alpha$ or capped).

n_win Number of entries with `q` in the window.

D_alpha_win Number of decoys in the window.

decoy_frac_win Decoy fraction in the window (`D_alpha_win/n_win`).

upper_used Upper endpoint actually used for the window.

truncated Logical indicating whether the requested window was truncated (or dropped) due to export limits.

This table is used to assess boundary support: very small `n_win` or `D_alpha_win` indicates an unstable/poorly-supported cutoff.

Examples

```
library(tibble)

set.seed(1)
n <- 5000
df <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  pep = NA_real_
)
df$q <- diagFDR:::tdc_qvalues(df$score, df$is_decoy, add_decoy = 1L)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy_tdc")

dfdr_local_tail(x, alphas = c(1e-3, 5e-3, 1e-2), win_rel = 0.2, truncation = "cap")
```

---

dfdr_pep_decoy_sanity *Decoy PEP sanity checks*

Description

Summarises how many decoys receive surprisingly small PEP values. Under a well-calibrated PEP, decoys should typically have high error probabilities; large fractions of decoys below small PEP thresholds can indicate issues with PEP calibration or labeling.

Usage

```
dfdr_pep_decoy_sanity(x, thresholds = c(0.01, 0.05, 0.1, 0.2))
```

Arguments

x	A dfdr_tbl with a non-missing pep column.
thresholds	Numeric vector of PEP thresholds in $(0, 1]$ to summarise.

Value

A tibble with one row per threshold. Columns include:

threshold The PEP cutoff used for counting.

n_decoy Number of decoys with finite PEP in $(0, 1]$.

n_target Number of targets with finite PEP in $(0, 1]$.

decoy_le Count of decoys with pep <= threshold.

target_le Count of targets with pep <= threshold.

frac_decoy_le decoy_le / n_decoy.

frac_target_le target_le / n_target.

Examples

```
library(tibble)

set.seed(1)
n <- 5000
df <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  q = pmin(1, rank(-score) / n),
  pep = stats::runif(n)
)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy")

dfdr_pep_decoy_sanity(x, thresholds = c(0.01, 0.05, 0.1))
```

---

dfdr_pep_reliability *Posterior Error Probability (PEP) reliability and internal PEP calibration error (IPE)*

Description

Bins identifications by predicted PEP and compares mean predicted PEP to the observed decoy fraction within each bin (internal target-decoy consistency check).

Usage

```
dfdr_pep_reliability(x, binwidth = 0.05, n_min = 200, pep_max = 0.5)
```

Arguments

x	A dfdr_tbl with a non-missing pep column.
binwidth	Numeric bin width for PEP binning (default 0.05).
n_min	Integer. Minimum bin size to include bins in the IPE summary.
pep_max	Numeric. Maximum mean PEP to include in the IPE summary (default 0.5).

Value

A list with components:

bins A tibble with one row per PEP bin and columns pep_mean (mean predicted PEP), decoy_rate (observed decoy fraction), and n (bin size).

IPE Numeric scalar. The internal PEP calibration error (IPE), computed as a weighted mean absolute deviation between pep_mean and decoy_rate across eligible bins (n >= n_min and pep_mean <= pep_max). NA if no eligible bins exist.

params A list of parameters used (binwidth, n_min, pep_max).

Examples

```
library(tibble)

set.seed(1)
n <- 5000
df <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  q = pmin(1, rank(-score) / n),         # simple monotone q-like values
  pep = pmin(1, pmax(0, stats::runif(n))) # toy PEP in [0,1]
)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy")

out <- dfdr_pep_reliability(x, binwidth = 0.1, n_min = 50, pep_max = 0.5)
out$IPE
head(out$bins)
```

---

dfdr_pep_reliability_tdc

Target-focused PEP reliability using a TDC-style error proxy

Description

Bins identifications by predicted PEP and estimates an error proxy for targets within each bin using decoys as a proxy via a target-decoy-competition (TDC) style ratio $(D + add\_decoy)/T$.

Usage

```
dfdr_pep_reliability_tdc(x, breaks = seq(0, 0.5, by = 0.05), add_decoy = 0L)
```

Arguments

x	A `dfdr_tbl` with columns `pep` and `is_decoy`.
breaks	Numeric vector of PEP bin edges.
add_decoy	Integer. Additive correction to the decoy count in each bin (default 0). Use 1 for a conservative small-sample correction.

Value

A tibble with one row per PEP bin, including:

bin Formatted bin label.

n_target Number of targets in the bin.

n_decoy Number of decoys in the bin.

pep_mean_targets Mean PEP among targets in the bin.

pep_mean_all Mean PEP among all entries in the bin.

err_hat_target Estimated target error proxy $(D + add\_decoy)/T$, capped at 1.

pep_bin_mid Midpoint of the PEP bin (useful for plotting).

Examples

```
library(tibble)

set.seed(1)
n <- 8000
df <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.9, 0.1)),
  score = rnorm(n),
  q = stats::runif(n),
  pep = stats::runif(n)
)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy")

dfdr_pep_reliability_tdc(x, breaks = seq(0, 0.5, by = 0.1), add_decoy = 1L)
```

`dfdr_pi0_storey` *Storey π_$0(\lambda)$ tail-uniformity diagnostic*

Description

Estimates $\pi_0(\lambda) = \#\{p > \lambda\}/((1-\lambda)m)$ over a grid of `lambdas`. This is a plausibility diagnostic for (pseudo-)p-values: under a well-calibrated null, the upper tail should be approximately uniform, leading to stable $\pi_0(\lambda)$ curves.

Usage

```
dfdr_pi0_storey(
  x,
  lambdas = seq(0.5, 0.95, by = 0.05),
  stratify = NULL,
  min_n = 2000,
  clamp = TRUE
)
```

Arguments

`x`	A `dfdr_tbl` (or data.frame) containing column `p`.
`lambdas`	Numeric vector in $[0, 1)$. Typical range: 0.5–0.95.

stratify	Optional character vector of column names used to stratify the diagnostic (e.g. c("run")).
min_n	Integer. Minimum number of finite p-values required per stratum.
clamp	Logical. If TRUE (default), clamp pi0_hat into [0,1] for reporting.

Value

A list with components:

pi0 A tibble with one row per lambda per stratum and columns stratum, lambda, pi0_hat, and n (the number of finite p-values in the stratum).

summary A tibble with one row per stratum, including n and summary statistics of pi0_hat across lambdas (median_pi0, sd_pi0, iqr_pi0). Strata with n < min_n are reported with NA metrics and a note.

Examples

```
library(tibble)

set.seed(1)
n <- 6000
df <- tibble(
  run = sample(c("run1", "run2"), n, replace = TRUE),
  # mostly uniform p-values + a small enriched component
  p = c(stats::runif(5400), stats::rbeta(600, 0.3, 1))
)

out <- dfdr_pi0_storey(df, stratify = "run", lambdas = seq(0.5, 0.9, by = 0.1), min_n = 500)
head(out$pi0)
out$summary
```

---

dfdr_plot_bh_elasticity

Plot BH elasticity (Jaccard) vs alpha

Description

Convenience plotting function for the output of dfdr_bh_elasticity. The x-axis is log10(alpha) and the y-axis is the Jaccard overlap between BH discovery sets at alpha and (1+eps)*alpha.

Usage

```
dfdr_plot_bh_elasticity(
  el_tbl,
  xlab = "alpha (log10)",
  title = "BH elasticity: Jaccard(alpha, (1+eps)alpha)"
)
```

Arguments

el_tbl	A tibble as returned by dfdr_bh_elasticity.
xlab	Character. X-axis label.
title	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

set.seed(1)
n <- 2000
x <- tibble(
  id = as.character(seq_len(n)),
  p = c(stats::runif(1800), stats::rbeta(200, 0.3, 1))
)

el <- dfdr_bh_elasticity(x, alphas = c(1e-3, 5e-3, 1e-2, 2e-2), eps = 0.2)
p <- dfdr_plot_bh_elasticity(el)
p
```

---

dfdr_plot_cv *Plot* CV_hat *versus* alpha

Description

Plots a stability proxy (CV_hat) against the threshold alpha for one or more lists.

Usage

```
dfdr_plot_cv(
  stab_tbl,
  xlab = "alpha (log10)",
  title = "Stability proxy CV_hat vs alpha"
)
```

Arguments

stab_tbl	A stability table (e.g. from dfdr_curve_stability) with columns alpha, CV_hat, and list.
xlab	Character. X-axis label.
title	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

stab_tbl <- tibble(
  list = rep(c("A", "B"), each = 4),
  alpha = rep(c(1e-3, 2e-3, 5e-3, 1e-2), times = 2),
  CV_hat = c(0.30, 0.22, 0.15, 0.12, 0.40, 0.28, 0.20, 0.18)
)
dfdr_plot_cv(stab_tbl)
```

---

dfdr_plot_dalpha *Plot decoy support D_α versus* alpha

Description

Plots the number of accepted decoys at each threshold `alpha`. This is a key stability indicator: very small D_α implies granular/unstable behaviour of target-decoy based estimates.

Usage

```
dfdr_plot_dalpha(
  stab_tbl,
  xlab = "alpha (log10)",
  title = "Decoy support D_alpha vs alpha"
)
```

Arguments

`stab_tbl`	A stability table (e.g. from `dfdr_curve_stability`) containing columns `alpha`, `D_alpha`, and `list`.
`xlab`	Character. X-axis label.
`title`	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

stab_tbl <- tibble(
  list = rep(c("A", "B"), each = 4),
  alpha = rep(c(1e-3, 2e-3, 5e-3, 1e-2), times = 2),
  D_alpha = c(5, 8, 20, 40, 2, 3, 6, 10)
)
dfdr_plot_dalpha(stab_tbl)
```

---

dfdr_plot_dwin *Plot local decoy support* D_alpha_win *versus* alpha

---

Description

Plots the number of decoys in a right-neighborhood above each threshold (as computed by dfdr_local_tail). This approximates how well supported the boundary is by nearby decoys.

Usage

```
dfdr_plot_dwin(
  dwin_tbl,
  xlab = "alpha (log10)",
  title = "Local decoy support D_alpha_win vs alpha"
)
```

Arguments

dwin_tbl	A local-tail table (e.g. from dfdr_local_tail) containing columns alpha, D_alpha_win, and list.
xlab	Character. X-axis label.
title	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

dwin_tbl <- tibble(
  list = rep(c("A", "B"), each = 4),
  alpha = rep(c(1e-3, 2e-3, 5e-3, 1e-2), times = 2),
  D_alpha_win = c(1, 2, 6, 15, 0, 1, 2, 4)
)
dfdr_plot_dwin(dwin_tbl)
```

`dfdr_plot_elasticity` *Plot threshold elasticity (Jaccard) versus* `alpha`

Description

Plots Jaccard overlap values (as returned by `dfdr_elasticity`) against `alpha`. Lower overlap indicates greater sensitivity of the accepted set to small changes of the threshold.

Usage

```
dfdr_plot_elasticity(
  el_tbl,
  xlab = "alpha (log10)",
  title = "Threshold elasticity (Jaccard) vs alpha"
)
```

Arguments

`el_tbl`	An elasticity table (e.g. from `dfdr_elasticity`) containing columns `alpha`, `jaccard`, and `list`.
`xlab`	Character. X-axis label.
`title`	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

el_tbl <- tibble(
  list = rep(c("A", "B"), each = 4),
  alpha = rep(c(1e-3, 2e-3, 5e-3, 1e-2), times = 2),
  jaccard = c(0.95, 0.93, 0.90, 0.88, 0.92, 0.90, 0.86, 0.82)
)
dfdr_plot_elasticity(el_tbl)
```

dfdr_plot_equal_chance *Plot equal-chance plausibility by q-value bands*

Description

Visualises decoy fractions by q-value band (as produced by dfdr_equal_chance_qbands). A horizontal reference at 0.5 indicates the expected decoy fraction under an "equal-chance" region assumption.

Usage

```
dfdr_plot_equal_chance(
  bands_tbl,
  title = "Equal-chance plausibility: decoy fraction by q-band"
)
```

Arguments

bands_tbl	A q-band table (e.g. out$bands from dfdr_equal_chance_qbands) containing columns q_mean, decoy_frac, and n.
title	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

bands_tbl <- tibble(
  q_mean = c(0.05, 0.15, 0.30, 0.45),
  decoy_frac = c(0.10, 0.20, 0.45, 0.52),
  n = c(2000, 1500, 800, 400)
)
dfdr_plot_equal_chance(bands_tbl)
```

`dfdr_plot_fdrhat_pi0` *Plot Storey-style estimated FDR curve: $\hat{\pi}\_0mt/R(t)$*

Description

Given p-values (or pseudo-p-values), estimates π_0 using `cp4p::estim.pi0()` and plots

$$\widehat{\mathrm{FDR}}(t) = \hat{\pi}_0 \, m \, t / R(t),$$

where $R(t) = \#\{p \le t\}$ and m is the number of finite p-values.

Usage

```
dfdr_plot_fdrhat_pi0(
  x,
  sel = c("all", "target", "decoy"),
  pi0.method = "pounds",
  nbins = 20,
  pz = 0.05,
  t_grid = 10^seq(-6, 0, length.out = 400),
  r_min = 1,
  cap_one = TRUE
)
```

Arguments

`x`	A `dfdr_tbl` (or data.frame) containing columns `id`, `p`, and `is_decoy`.
`sel`	Character. Subset to use: `"all"`, `"target"`, or `"decoy"`.
`pi0.method`	Character. Method passed to `cp4p::estim.pi0()` (not `"ALL"`).
`nbins`	Integer. Passed to `cp4p::estim.pi0()`.
`pz`	Numeric. Passed to `cp4p::estim.pi0()`.
`t_grid`	Numeric vector of thresholds in $(0, 1]$ at which to evaluate the curve.
`r_min`	Numeric. Minimum value used to determine when to compute the curve to avoid division by zero at tiny `t`.
`cap_one`	Logical. If `TRUE`, caps `fdr_hat` at 1.

Value

A list with components:

plot A ggplot object.

data A tibble with columns `t`, `R`, and `fdr_hat`.

pi0_hat Numeric scalar $\hat{\pi}_0$.

m Integer. Number of finite p-values used.

Examples

```
library(tibble)

if (requireNamespace("cp4p", quietly = TRUE)) {
  set.seed(1)
  n <- 4000
  df <- tibble(
    id = as.character(seq_len(n)),
    is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
    p = c(stats::runif(3600), stats::rbeta(400, 0.3, 1))
  )
  out <- dfdr_plot_fdrhat_pi0(df, sel = "all", pi0.method = "pounds", nbins = 10)
  out$pi0_hat
  out$plot
}
```

`dfdr_plot_pep_density_by_decoy` *Plot PEP density by decoy/target*

Description

Plots density curves of PEP values for targets vs decoys, truncated to `pep <= pep_max`.

Usage

```
dfdr_plot_pep_density_by_decoy(
  x,
  pep_max = 0.5,
  bw = "nrd0",
  adjust = 1,
  title = "PEP distribution (density): targets vs decoys"
)
```

Arguments

`x`	A `dfdr_tbl` with a `pep` column.
`pep_max`	Numeric. Maximum PEP displayed (default 0.5).
`bw`	Bandwidth passed to `ggplot2::geom_density()`.
`adjust`	Numeric smoothing adjustment passed to `geom_density()`.
`title`	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

set.seed(1)
n <- 4000
df <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  q = stats::runif(n, 0, 0.05),
  pep = stats::runif(n)
)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy")
dfdr_plot_pep_density_by_decoy(x, pep_max = 0.5)
```

---

dfdr_plot_pep_reliability

Plot PEP reliability

Description

Visualises PEP reliability by plotting the observed decoy fraction against the mean predicted PEP in each bin (from `dfdr_pep_reliability`). Point size reflects bin counts.

Usage

```
dfdr_plot_pep_reliability(bins_tbl, title = "PEP reliability")
```

Arguments

`bins_tbl`	A PEP reliability bins table (e.g. `out$bins` from `dfdr_pep_reliability`) containing columns `pep_mean`, `decoy_rate`, and `n`.
`title`	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

bins_tbl <- tibble(
  pep_mean = c(0.05, 0.15, 0.25, 0.35),
  decoy_rate = c(0.06, 0.14, 0.30, 0.40),
  n = c(500, 400, 250, 120)
```

```
)
dfdr_plot_pep_reliability(bins_tbl)
```

`dfdr_plot_pep_reliability_tdc` *Plot target-focused PEP reliability (TDC-style)*

Description

Visualises the output of `dfdr_pep_reliability_tdc` by plotting the estimated target error proxy `err_hat_target` against the mean predicted target PEP per bin. Point size reflects the number of targets per bin.

Usage

```
dfdr_plot_pep_reliability_tdc(
  rel_tbl,
  title = "PEP reliability (targets; TDC-style D/T error proxy)"
)
```

Arguments

`rel_tbl`	Output of `dfdr_pep_reliability_tdc`.
`title`	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

rel_tbl <- tibble(
  bin = c("(0,0.1]", "(0.1,0.2]", "(0.2,0.3]"),
  n_target = c(800, 500, 200),
  n_decoy = c(20, 30, 40),
  pep_mean_targets = c(0.05, 0.15, 0.25),
  pep_mean_all = c(0.06, 0.17, 0.28),
  err_hat_target = c(0.03, 0.06, 0.20),
  pep_bin_mid = c(0.05, 0.15, 0.25)
)
dfdr_plot_pep_reliability_tdc(rel_tbl)
```

`dfdr_plot_pi0` *Plot Storey π_0(λ) curve*

Description

Plots $\pi_0(\lambda)$ estimates returned by `dfdr_pi0_storey`. Values closer to 1 suggest many nulls (few true effects), while values far below 1 (especially if unstable across `lambda`) may indicate deviations from tail-uniformity.

Usage

```
dfdr_plot_pi0(pi0_tbl, title = "Storey pi0(lambda) diagnostic")
```

Arguments

`pi0_tbl`	A tibble, typically `out$pi0` from `dfdr_pi0_storey`. Must contain columns `stratum`, `lambda`, and `pi0_hat`.
`title`	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

set.seed(1)
df <- tibble(
  stratum = rep("all", 5),
  lambda = seq(0.5, 0.9, by = 0.1),
  pi0_hat = c(0.95, 0.96, 0.97, 0.98, 0.99),
  n = 1000
)
p <- dfdr_plot_pi0(df)
p
```

`dfdr_plot_p_calibration`

Plot p-value calibration (ECDF minus uniform)

Description

Plots $\hat{F}(u) - u$ against u using the `ecdf` component returned by `dfdr_p_calibration`. Values above zero indicate potential inflation (excess of small p-values) relative to uniform.

Usage

```
dfdr_plot_p_calibration(ecdf_tbl, title = "P-value calibration: ECDF(p) - u")
```

Arguments

ecdf_tbl	A tibble, typically out$ecdf from dfdr_p_calibration. Must contain columns stratum, u, and Fhat.
title	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

set.seed(1)
n <- 2000
df <- tibble(
  id = as.character(seq_len(n)),
  run = sample(c("run1", "run2"), n, replace = TRUE),
  p = c(stats::runif(1800), stats::rbeta(200, 0.3, 1))
)
cal <- dfdr_p_calibration(df, stratify = "run", min_n = 100)
p <- dfdr_plot_p_calibration(cal$ecdf)
p
```

---

dfdr_plot_p_calibration2

P-value calibration plot (cp4p-style), with multiple pi0 reference curves

---

Description

Plots the ECDF of $1 - p$ and overlays reference curves derived from cp4p::estim.pi0() using multiple π_0 estimation methods. This provides a visual plausibility check for p-value calibration.

Usage

```
dfdr_plot_p_calibration2(
  x,
  sel = c("all", "decoy", "target"),
  nbins = 20,
  pz = 0.05,
  step = 0.001,
  return_data = FALSE
)
```

Arguments

x	A dfdr_tbl (or data.frame) containing columns id, is_decoy, and p.
sel	Character. Subset to use: "all", "decoy", or "target".
nbins	Integer. Passed to cp4p::estim.pi0().
pz	Numeric. Passed to cp4p::estim.pi0().
step	Numeric. Grid step size for abs = 1 - p.
return_data	Logical. If TRUE, return both the plot and the computed data used to build it.

Value

If return_data = FALSE (default), returns a ggplot object.

If return_data = TRUE, returns a list with components:

plot A ggplot object.

pi0 Named numeric vector of π_0 estimates (one per method).

data A list with main (ECDF data) and ref (reference curve data).

Examples

```
library(tibble)

set.seed(1)
n <- 2000
df <- tibble(
  id = as.character(seq_len(n)),
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.9, 0.1)),
  p = c(stats::runif(1800), stats::rbeta(200, 0.3, 1))
)

if (requireNamespace("cp4p", quietly = TRUE)) {
  g <- dfdr_plot_p_calibration2(df, sel = "all", return_data = FALSE)
  g
}
```

---

dfdr_plot_p_density_by_decoy

Plot (pseudo-)p-value density by decoy/target

Description

Plots density curves of (pseudo-)p-values for targets vs decoys. Optionally plots $-\log_{10}(p)$ to emphasise small p-values.

Usage

```
dfdr_plot_p_density_by_decoy(
  x,
  p_max = 1,
  bw = "nrd0",
  adjust = 1,
  title = "(Pseudo-)p-value distribution (density): targets vs decoys",
  log10_x = FALSE,
  eps = 1e-300
)
```

Arguments

x	A dfdr_tbl with columns p and is_decoy.
p_max	Numeric. Maximum p-value displayed (default 1).
bw	Bandwidth passed to ggplot2::geom_density().
adjust	Numeric smoothing adjustment passed to geom_density().
title	Character. Plot title.
log10_x	Logical. If TRUE, plot $-\log_{10}(p)$ instead of p.
eps	Numeric. Lower bound used to avoid Inf when log10_x = TRUE.

Value

A ggplot object.

Examples

```
library(tibble)

set.seed(1)
n <- 4000
df <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  q = stats::runif(n, 0, 0.05),
  pep = NA_real_,
  p = stats::runif(n)
)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy", p_source = "toy")
dfdr_plot_p_density_by_decoy(x, log10_x = TRUE)
```

`dfdr_plot_scope_disagreement_matrix` *Plot scope disagreement as a Jaccard overlap heatmap*

Description

Computes accepted target ID sets for each list at a fixed `alpha` and visualises pairwise Jaccard overlaps as a heatmap. IDs are compared at the "base" level by stripping any `"run||"` prefix.

Usage

```
dfdr_plot_scope_disagreement_matrix(xs, alpha = 0.01)
```

Arguments

`xs`	Named list of `dfdr_tbl` objects.
`alpha`	Numeric threshold used to define accepted targets (`q <= alpha`).

Value

A ggplot object (heatmap).

Examples

```
library(tibble)

set.seed(1)
n <- 2000

make_x <- function(label) {
  df <- tibble(
    id = paste0("run1||", as.character(seq_len(n))),
    run = "run1",
    is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
    score = rnorm(n),
    q = stats::runif(n, 0, 0.05),
    pep = NA_real_
  )
  as_dfdr_tbl(df, unit = "psm", scope = label, q_source = "toy")
}

xs <- list(A = make_x("A"), B = make_x("B"), C = make_x("C"))
dfdr_plot_scope_disagreement_matrix(xs, alpha = 0.01)
```

`dfdr_plot_score_distributions` *Plot target vs decoy score distributions*

Description

Plots density distributions of scores (if available) or q-values (fallback) for targets vs decoys. This provides a quick sanity check that decoys are shifted toward lower scores (or higher q-values).

Usage

```
dfdr_plot_score_distributions(x, title = "Target vs Decoy Score Distributions")
```

Arguments

`x`	A `dfdr_tbl`.
`title`	Character. Plot title.

Value

A ggplot object.

Examples

```
library(tibble)

set.seed(1)
n <- 4000
df <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  q = pmin(1, rank(-score) / n),
  pep = NA_real_
)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy")
dfdr_plot_score_distributions(x)
```

`dfdr_p_calibration` *P-value calibration diagnostic (ECDF vs uniform; stratified)*

Description

Computes the empirical CDF $\hat{F}(u) = P(p \leq u)$ on a grid of u values and summarises departures from uniformity in a decision-relevant region $u \leq u_{\max}$. Intended as a plausibility diagnostic for (pseudo-)p-values (e.g. under the null, $\hat{F}(u) \approx u$).

Usage

```
dfdr_p_calibration(
  x,
  u_grid = c(seq(0.001, 0.1, by = 0.001), seq(0.11, 1, by = 0.01)),
  u_max = 0.1,
  stratify = NULL,
  min_n = 200
)
```

Arguments

`x`	A `dfdr_tbl` (or data.frame) containing columns `id` and `p`.
`u_grid`	Numeric vector in $(0, 1]$. Grid of `u` values for evaluating the ECDF.
`u_max`	Numeric scalar in $(0, 1]$. Upper bound of `u` used for inflation summaries (default 0.1).
`stratify`	Optional character vector of column names used to stratify diagnostics (e.g. `c("run")`).
`min_n`	Integer. Minimum number of finite p-values required per stratum.

Value

A list with components:

ecdf A tibble with columns `stratum`, `u`, `Fhat`, and `n`. There is one row per `u` value per stratum.

summary A tibble with one row per stratum, including `n` (number of finite p-values), `max_inflation` (maximum of `Fhat(u)-u` for `u <= u_max`), and `auc_inflation` (area under the positive part of `Fhat(u)-u` on $[0, u_{\max}]$, normalised by `u_max`). Strata with `n < min_n` are reported with `NA` metrics and a `note`.

Examples

```
library(tibble)

set.seed(1)
n <- 5000
df <- tibble(
  id = as.character(seq_len(n)),
```

```
  run = sample(c("run1", "run2"), n, replace = TRUE),
  # mostly uniform p-values + a small enriched component
  p = c(stats::runif(4500), stats::rbeta(500, 0.3, 1))
)

out <- dfdr_p_calibration(df, stratify = "run", u_max = 0.1, min_n = 200)
head(out$ecdf)
out$summary
```

`dfdr_render_report` *Render a human-readable HTML report from* `dfdr_run_all` *output*

Description

Renders an HTML report that summarises key diagnostics and embeds plots contained in `diag$plots`. The report is intended for interactive review and verifiable reporting.

Usage

```
dfdr_render_report(
  diag,
  out_dir,
  filename = "diagFDR_report.html",
  self_contained = FALSE,
  open = FALSE
)
```

Arguments

`diag`	A list as returned by `dfdr_run_all`.
`out_dir`	Character scalar. Output directory (created if it does not exist).
`filename`	Character scalar. Output HTML filename (default `"diagFDR_report.html"`).
`self_contained`	Logical. If `TRUE`, embed resources into a single HTML file (may be larger). Default `FALSE`.
`open`	Logical. If `TRUE`, open the report in the default browser after rendering.

Details

This function requires the **rmarkdown** package (suggested dependency) and uses the built-in R Markdown template shipped with the package (`inst/templates/dfdr_report.Rmd`).

Value

The path to the rendered HTML file, returned invisibly. The function is called for its side effect of creating an HTML report on disk.

Examples

```
# A minimal example that renders a report from a toy dataset.
# This example is conditional because rmarkdown is in Suggests.
if (requireNamespace("rmarkdown", quietly = TRUE)) {
  library(tibble)
  tmpdir <- tempdir()

  set.seed(1)
  n <- 3000
  df <- tibble(
    id = as.character(seq_len(n)),
    run = "run1",
    is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
    score = rnorm(n),
    q = pmin(1, rank(-score) / n),
    pep = NA_real_
  )
  x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy")

  diag <- dfdr_run_all(
    xs = list(toy = x),
    alpha_main = 0.01,
    compute_pseudo_pvalues = FALSE
  )

  # Render to a temporary directory (does not open a browser during checks)
  dfdr_render_report(diag, out_dir = tmpdir, open = FALSE)
}
```

`dfdr_run_all` *Run a standard set of FDR QC diagnostics*

Description

Runs stability, boundary-support, and plausibility diagnostics on a named list of `dfdr_tbl` objects and returns both summary tables and plots.

Usage

```
dfdr_run_all(
  xs,
  alpha_main = 0.01,
  alphas = c(1e-04, 5e-04, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2),
  eps = 0.2,
  win_rel = 0.2,
  truncation = "warn_drop",
  k_fixed = 10,
  k_sqrt_mult = 2,
```

```
  qband_breaks = c(0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5),
  low_conf = c(0.2, 0.5),
  min_N_equalchance = 2000,
  pep_binwidth = 0.05,
  pep_n_min = 200,
  pep_max = 0.5,
  pep_sanity_thresholds = c(0.01, 0.05, 0.1, 0.2),
  pep_density_max = 0.5,
  pep_rel_tdc_breaks = seq(0, 0.5, by = 0.05),
  pep_rel_tdc_add_decoy = 0L,
  compute_pseudo_pvalues = TRUE,
  pseudo_pvalue_method = "decoy_ecdf",
  p_u_grid = c(seq(0.001, 0.1, by = 0.001), seq(0.11, 1, by = 0.01)),
  p_u_max = 0.1,
  p_stratify = NULL,
  p_min_n_calib = 200,
  p_lambdas = seq(0.5, 0.95, by = 0.05),
  p_min_n_pi0 = 2000,
  bh_boundary = c("mult", "add"),
  bh_win = 0.2,
  bh_delta = NULL
)
```

Arguments

xs	Named list of dfdr_tbl objects.
alpha_main	Numeric scalar in $(0, 1]$. Headline threshold (e.g. 0.01).
alphas	Numeric vector of thresholds in $(0, 1]$ used to build curves.
eps	Numeric scalar. Relative perturbation used in elasticity (e.g. 0.2).
win_rel	Numeric scalar. Relative window width for local tail support (e.g. 0.2).
truncation	Character. Truncation policy for local tail windows; see dfdr_local_tail.
k_fixed	Integer. Fixed +/-K used in sensitivity intervals.
k_sqrt_mult	Numeric. Multiplier for adaptive +/-$\lceil mult\sqrt{D} \rceil$ intervals.
qband_breaks	Numeric vector. Cutpoints for q-band equal-chance diagnostic.
low_conf	Length-2 numeric vector. Pooled low-confidence region for the equal-chance test.
min_N_equalchance	Integer. Minimum pooled N for equal-chance test.
pep_binwidth	Numeric. Bin width for PEP reliability bins.
pep_n_min	Integer. Minimum bin size to include PEP bins in IPE summary/plots.
pep_max	Numeric. Max mean PEP included in IPE summary.
pep_sanity_thresholds	Numeric vector. Thresholds for decoy PEP sanity summaries.
pep_density_max	Numeric. Max PEP displayed in target/decoy density plot.

`pep_rel_tdc_breaks`
: Numeric vector. Breaks used for the TDC-style PEP reliability bins.

`pep_rel_tdc_add_decoy`
: Integer. Additive correction used in D/T for TDC-style reliability.

`compute_pseudo_pvalues`
: Logical. If `TRUE`, attempt to compute pseudo-p-values when `p` is missing.

`pseudo_pvalue_method`
: Character. Method passed to `score_to_pvalue` (default `"decoy_ecdf"`).

`p_u_grid`
: Numeric vector in $(0, 1]$. Grid for p-value ECDF calibration diagnostics.

`p_u_max`
: Numeric scalar in $(0, 1]$. Upper bound of u used for p-value inflation summaries.

`p_stratify`
: Optional character vector of columns for stratified p-value diagnostics.

`p_min_n_calib`
: Integer. Minimum n for p-value calibration per stratum.

`p_lambdas`
: Numeric vector in $[0, 1)$. Lambdas for Storey $\pi_0(\lambda)$ diagnostic.

`p_min_n_pi0`
: Integer. Minimum n for $\pi_0(\lambda)$ diagnostic per stratum.

`bh_boundary`
: Character. Boundary window type for BH diagnostics (`"mult"` or `"add"`).

`bh_win`
: Numeric. Relative window size for BH boundary support when `bh_boundary="mult"`.

`bh_delta`
: Numeric. Additive window size when `bh_boundary="add"`.

Details

Diagnostics include (depending on available columns):

- Target/decoy headline diagnostics at `alpha_main` (counts, FDR estimate, stability proxies)
- Stability curves across `alphas`
- Local tail decoy support near each `alpha`
- Elasticity (Jaccard overlap under threshold perturbation)
- PEP reliability / IPE and PEP sanity summaries (if `pep` is present)
- Equal-chance by q-bands (always attempted; may be not applicable if q-range is truncated)
- (Pseudo-)p-value calibration and Storey $\pi_0(\lambda)$ (if `p` is present or computed)
- BH diagnostics (if `p` is present)

Value

A list with components:

tables A named list of result tables (tibbles) for the computed diagnostics (e.g. `headline`, `stability`, `local_tail`, `elasticity`, and optional PEP/p-value/BH-related tables).

plots A named list of ggplot objects produced from the diagnostics.

objects The input `xs`, possibly augmented with computed pseudo-p-values if `compute_pseudo_pvalues=TRUE`.

params A list of parameter values used to compute the diagnostics.

Examples

```
library(tibble)

set.seed(1)
n <- 2000
df <- tibble(
  id = as.character(seq_len(n)),
  run = sample(c("run1", "run2"), n, replace = TRUE),
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  # Toy q/pep/p for demonstration purposes
  q = pmin(1, rank(-score) / n),
  pep = stats::runif(n),
  p = stats::runif(n)
)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy", p_source = "toy")

diag <- dfdr_run_all(
  xs = list(toy = x),
  alpha_main = 0.01,
  compute_pseudo_pvalues = FALSE
)

names(diag$tables)
names(diag$plots)
diag$tables$headline
```

---

dfdr_run_jaccard *Inter-run Jaccard overlap matrix at a fixed threshold*

Description

Computes pairwise Jaccard similarity between runs based on accepted target IDs at a fixed threshold `alpha`. This can reveal run-to-run disagreement or heterogeneity in identifications.

Usage

```
dfdr_run_jaccard(x, alpha = 0.01)
```

Arguments

`x`	An `dfdr_tbl` with a non-missing `run` column.
`alpha`	Numeric FDR threshold in $(0, 1]$ used to define accepted targets (`q <= alpha`).

Value

A tibble with one row per run pair and columns:

run1 Name/identifier of the first run.

run2 Name/identifier of the second run.

jaccard Jaccard similarity between accepted target ID sets.

Examples

```
library(tibble)

set.seed(1)
n <- 8000
df <- tibble(
  id = as.character(seq_len(n)),
  run = sample(c("runA", "runB", "runC"), n, replace = TRUE),
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  pep = NA_real_
)
df$q <- diagFDR:::tdc_qvalues(df$score, df$is_decoy, add_decoy = 1L)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy_tdc")

dfdr_run_jaccard(x, alpha = 0.01)
```

---

dfdr_scope_disagreement

Scope disagreement between two lists

Description

Computes the Jaccard overlap of accepted target IDs between two `dfdr_tbl` objects across a set of thresholds. This is useful for comparing results obtained under different FDR scopes (e.g. run-wise vs global) or from different pipelines, provided that both inputs use compatible `id` semantics.

Usage

```
dfdr_scope_disagreement(
  x1,
  x2,
  alphas,
  label1 = "A",
  label2 = "B",
  normalize_ids = TRUE
)
```

Arguments

x1	First dfdr_tbl.
x2	Second dfdr_tbl.
alphas	Numeric vector of thresholds in $(0, 1]$.
label1	Character label for x1 in the output (default "A").
label2	Character label for x2 in the output (default "B").
normalize_ids	Logical. If TRUE (default), extracts a "base" ID by stripping any run prefix of the form "<run>||<id>" (everything up to and including "||"). Set to FALSE if IDs are already comparable between x1 and x2.

Value

A tibble with one row per alpha and columns:

alpha The threshold.

label1,label2 The labels identifying the two inputs.

n1,n2 Numbers of accepted target IDs in x1 and x2 at alpha.

jaccard Jaccard similarity between the two accepted target ID sets.

Examples

```
library(tibble)

set.seed(1)
n <- 4000

# Two "lists" with slightly different q-values for the same base IDs
base_ids <- as.character(seq_len(n))

df1 <- tibble(
  id = paste0("run1||", base_ids),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  q = stats::runif(n, 0, 0.05),
  pep = NA_real_
)

df2 <- tibble(
  id = base_ids,              # no run prefix here
  run = "all",
  is_decoy = df1$is_decoy,    # same decoy labels for simplicity
  score = df1$score + rnorm(n, sd = 0.2),
  q = pmin(1, df1$q * 1.2),   # slightly perturbed q-values
  pep = NA_real_
)

x1 <- as_dfdr_tbl(df1, unit = "psm", scope = "runwise", q_source = "toy")
x2 <- as_dfdr_tbl(df2, unit = "psm", scope = "global",  q_source = "toy")
```

```
dfdr_scope_disagreement(x1, x2, alphas = c(0.005, 0.01, 0.02), normalize_ids = TRUE)
```

`dfdr_summary_headline` *Build a headline summary table (one row per list)*

Description

Convenience helper to assemble a compact, export-ready summary table from the output of `dfdr_run_all`. The function merges headline metrics with selected columns from other diagnostics (e.g. local tail support, equal-chance pooled estimate, and PEP reliability headline if available).

Usage

```
dfdr_summary_headline(diag)
```

Arguments

`diag`	A list as returned by `dfdr_run_all`.

Value

A tibble with one row per element of the input list used in `dfdr_run_all` (i.e. one row per `list` label). The table is suitable for exporting to a CSV (e.g. `summary_headline.csv`).

The output typically contains (when available) headline target/decoy counts at `alpha_main`, local boundary-support columns (e.g. `D_alpha_win`, `n_win`), equal-chance pooled columns (e.g. `pi_D_hat`, `effect_abs`), and PEP reliability headline quantities (e.g. `IPE`). If `diag$tables$headline` is missing or empty, an empty tibble is returned.

Examples

```
library(tibble)

set.seed(1)
n <- 4000
df <- tibble(
  id = as.character(seq_len(n)),
  run = sample(c("run1", "run2"), n, replace = TRUE),
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n)
)
df$q <- diagFDR:::tdc_qvalues(df$score, df$is_decoy, add_decoy = 1L)
df$pep <- NA_real_
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy_tdc")

diag <- dfdr_run_all(
  xs = list(toy = x),
  alpha_main = 0.01,
```

```
  compute_pseudo_pvalues = FALSE
)

dfdr_summary_headline(diag)
```

`dfdr_sumpep` *Expected number of false targets among accepted identifications*

Description

Computes $\sum_{i \in A(\alpha)} \mathrm{PEP}_i$ among accepted targets for each threshold α, where acceptance is defined by `q <= alpha`.

Usage

```
dfdr_sumpep(x, alphas)
```

Arguments

`x`	An `dfdr_tbl` with a non-missing `pep` column.
`alphas`	Numeric vector of thresholds in $(0, 1]$.

Value

A tibble with one row per `alpha`. Columns include:

alpha The threshold.

n_targets Number of accepted targets (`!is_decoy` and `q <= alpha`).

sum_PEP Sum of PEP values among accepted targets (expected false targets).

mean_PEP Mean PEP among accepted targets.

Examples

```
library(tibble)

set.seed(1)
n <- 5000
df <- tibble(
  id = as.character(seq_len(n)),
  run = "run1",
  is_decoy = sample(c(FALSE, TRUE), n, replace = TRUE, prob = c(0.95, 0.05)),
  score = rnorm(n),
  q = pmin(1, rank(-score) / n),
  pep = stats::runif(n)
)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy")

dfdr_sumpep(x, alphas = c(0.001, 0.01, 0.05))
```

`dfdr_write_manifest` *Write a manifest file describing parameters, list metadata, and warnings*

Description

Writes a manifest alongside exported outputs to record key run parameters and per-list metadata (unit, scope, q-source, export ceiling, etc.). This helps tie diagnostic outputs to the exact inputs and settings used.

Usage

```
dfdr_write_manifest(diag, out_dir, filename = "manifest")
```

Arguments

`diag`	A list as returned by `dfdr_run_all`.
`out_dir`	Character scalar. Output directory (created if it does not exist).
`filename`	Character scalar. Base filename without extension (default `"manifest"`).

Details

If **jsonlite** is available (suggested dependency), the manifest is written as JSON (`*.json`); otherwise a human-readable plain text file (`*.txt`) is written.

Value

The path to the manifest file, returned invisibly. The function is called for its side effect of writing a file to disk.

Examples

```
library(tibble)

# Create a minimal dfdr_run_all-like object to demonstrate manifest writing
df <- tibble(
  id = c("1","2","3"),
  run = "run1",
  is_decoy = c(FALSE, TRUE, FALSE),
  score = c(10, 9, 8),
  q = c(0.01, 0.02, 0.03),
  pep = NA_real_
)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy")

diag <- list(
  objects = list(toy = x),
  params = list(alpha_main = 0.01),
```

```
  warnings = character()
)

out_dir <- tempdir()
dfdr_write_manifest(diag, out_dir = out_dir, filename = "manifest_example")
```

`dfdr_write_readme` *Write a human-readable* `README.md` *report*

Description

Produces a lightweight narrative report in Markdown that summarises the most important diagnostics (headline summary, key plots, exported tables) and records per-list metadata (unit, scope, q-source, etc.) from a `dfdr_run_all` result. This is intended to accompany an export folder for auditing and sharing.

Usage

```
dfdr_write_readme(diag, out_dir, filename = "README.md")
```

Arguments

`diag`	A list as returned by `dfdr_run_all`.
`out_dir`	Character scalar. Output directory (created if it does not exist).
`filename`	Character scalar. Markdown filename (default `"README.md"`).

Value

The path to the written README file, returned invisibly. The function is called for its side effect of writing a file to disk.

Examples

```
library(tibble)

# Minimal dfdr_run_all-like object for demonstrating README writing
df <- tibble(
  id = c("1","2","3"),
  run = "run1",
  is_decoy = c(FALSE, TRUE, FALSE),
  score = c(10, 9, 8),
  q = c(0.01, 0.02, 0.03),
  pep = NA_real_
)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy")

diag <- list(
  objects = list(toy = x),
```

```
  params = list(alpha_main = 0.01),
  tables = list(headline = tibble(list = "toy", alpha = 0.01)),
  warnings = character()
)

out_dir <- tempdir()
dfdr_write_readme(diag, out_dir = out_dir, filename = "README_example.md")
```

`dfdr_write_report` *Write diagnostic outputs to a folder*

Description

Writes tables as CSV and plots as PNG, optionally creates a PPTX (requires officer+rvg).

Exports the output of `dfdr_run_all` to disk. Depending on selected `formats`, this function can:

- write diagnostic tables as CSV;
- write plots as PNG;
- create a PowerPoint (`.pptx`) containing the plots (requires **officer** and **rvg** in Suggests);
- write a lightweight manifest (`dfdr_write_manifest`);
- write a human-readable `README.md` (`dfdr_write_readme`);
- write a compact headline summary table (`summary_headline.csv`).

Usage

```
dfdr_write_report(
  diag,
  out_dir,
  formats = c("csv", "png", "manifest", "readme", "summary"),
  pptx_path = file.path(out_dir, "diagFDR_report.pptx"),
  width = 7.2,
  height = 4.4,
  dpi = 180
)

dfdr_write_report(
  diag,
  out_dir,
  formats = c("csv", "png", "manifest", "readme", "summary"),
  pptx_path = file.path(out_dir, "diagFDR_report.pptx"),
  width = 7.2,
  height = 4.4,
  dpi = 180
)
```

Arguments

diag	A list as returned by dfdr_run_all.
out_dir	Character scalar. Output directory (created if it does not exist).
formats	Character vector. Subset of c("csv","png","pptx","manifest","readme","summary").
pptx_path	Character scalar. Output path for the PPTX file (only used if "pptx" is included in formats).
width, height	Numeric. Plot size (in inches) passed to ggplot2::ggsave() when writing PNGs.
dpi	Numeric. DPI passed to ggplot2::ggsave().

Value

Invisibly returns TRUE.

Returns TRUE invisibly. The function is called for its side effects of writing files to out_dir.

Examples

```
library(tibble)

# Create a minimal diag object with one table and one plot
diag <- list(
  tables = list(headline = tibble(list = "toy", alpha = 0.01, D_alpha = 10)),
  plots = list(example_plot = ggplot2::ggplot(tibble(x = 1:3, y = 1:3),
                                              ggplot2::aes(x, y)) +
               ggplot2::geom_point()),
  objects = list(),
  params = list(alpha_main = 0.01),
  warnings = character()
)

out_dir <- tempdir()
dfdr_write_report(
diag, out_dir = out_dir,
formats = c("csv", "png", "summary", "manifest", "readme"))
```

---

diann_global_minrunq *DIA-NN: global precursor list built by minimum run-wise q across runs*

Description

Constructs a precursor-level table by taking the minimum run-wise q-value across runs for each Precursor.Id. This mirrors a common scope misuse (aggregating run-wise q-values into a single global list) and is useful for scope-disagreement diagnostics.

Usage

```
diann_global_minrunq(
  rep,
  run_col = NULL,
  q_col = "Q.Value",
  pep_col = NULL,
  score_col = NULL,
  q_max_export = NA_real_,
  unit = "precursor",
  scope = "aggregated",
  q_source = paste0("min_run(", q_col, ")")
)
```

Arguments

rep	A DIA-NN report table (e.g. returned by read_diann_parquet).
run_col	Optional character. Name of the run column. If NULL, attempts to detect it (e.g. "Run" or "File.Name").
q_col	Character. Name of the q-value column to aggregate (default "Q.Value").
pep_col	Optional character. Name of the PEP column (default "PEP" if present).
score_col	Optional character. Name of a score column to carry along (if present).
q_max_export	Optional numeric export ceiling used by the tool (e.g. 0.5), stored as metadata for truncation-aware diagnostics.
unit	Character. Unit metadata stored in the returned object.
scope	Character. Scope metadata stored in the returned object.
q_source	Character. Source label stored in metadata.

Value

A dfdr_tbl with one row per precursor, where q equals the minimum of the run-wise q-values across runs for that precursor. The returned object is intended for diagnostics (e.g. scope disagreement), not as a recommended FDR procedure.

Examples

```
library(tibble)

rep <- tibble(
  Run = c("r1", "r2", "r1", "r2"),
  Precursor.Id = c("P1", "P1", "P2", "P2"),
  Decoy = c(0L, 0L, 0L, 1L),
  Q.Value = c(0.02, 0.01, 0.05, 0.04),
  PEP = c(0.03, 0.02, 0.1, 0.9)
)

x <- diann_global_minrunq(rep, run_col = "Run", q_col = "Q.Value", q_max_export = 0.5)
x
```

diann_global_precursor *DIA-NN -> global precursor universe (deduplicated by* `Precursor.Id`*)*

Description

Constructs a precursor-level universe by aggregating a DIA-NN report table to one row per `Precursor.Id`. For each precursor, the minimum q-value is used (via `safe_min`) and `is_decoy` is set to `TRUE` if any row in the group is a decoy.

Usage

```
diann_global_precursor(
  rep,
  q_col = "Q.Value",
  pep_col = NULL,
  score_col = NULL,
  q_max_export = NA_real_,
  unit = "precursor",
  scope = "global",
  q_source = q_col
)
```

Arguments

`rep`	A DIA-NN report tibble (e.g. returned by `read_diann_parquet`).
`q_col`	Character. Column name for q-values (default `"Q.Value"`).
`pep_col`	Optional character. PEP column name (default `"PEP"` if present).
`score_col`	Optional character. Score column name to retain; if missing, `score` is `NA`.
`q_max_export`	Optional numeric export ceiling used in DIA-NN export (e.g. 0.5).
`unit`	Character. Unit metadata stored in the returned object.
`scope`	Character. Scope metadata stored in the returned object.
`q_source`	Character. Label stored in metadata describing the q-value source.

Value

A `dfdr_tbl` (tibble subclass) with one row per precursor and required columns `id`, `is_decoy`, `q`, `pep`, `run`, `score`. Metadata are stored in `attr(x, "meta")`.

Examples

```
library(tibble)

rep <- tibble(
  Precursor.Id = c("P1", "P1", "P2"),
  Decoy = c(0L, 0L, 1L),
  Q.Value = c(0.01, 0.02, 0.03),
  PEP = c(0.02, 0.01, 0.9)
)

x <- diann_global_precursor(rep, q_col = "Q.Value", q_max_export = 0.5)
x
```

---

diann_runxprecursor *DIA-NN -> run-by-precursor universe (deduplicated within run)*

Description

Constructs a run-specific universe by aggregating a DIA-NN report table to one row per (`run`, `Precursor.Id`). The run column is detected automatically unless provided. For each group, the minimum q-value is used and `is_decoy` is `TRUE` if any row is a decoy.

Usage

```
diann_runxprecursor(
  rep,
  q_col = "Q.Value",
  run_col = NULL,
  pep_col = NULL,
  score_col = NULL,
  q_max_export = NA_real_,
  id_mode = c("id", "runxid"),
  unit = "runxprecursor",
  scope = "runwise",
  q_source = q_col
)
```

Arguments

`rep`	A DIA-NN report tibble.
`q_col`	Character. q-value column name (default `"Q.Value"`).
`run_col`	Optional character. Name of the run column. If `NULL`, the function attempts to detect one of `"Run"`, `"File.Name"`, or `"File.Name.Index"`.
`pep_col`	Optional character. PEP column name (default `"PEP"` if present).
`score_col`	Optional character. Score column name (if present).

q_max_export	Optional numeric export ceiling (e.g. 0.5), stored as metadata.
id_mode	Character. Either "runxid" (default) to set id = paste(run, Precursor.Id, sep="||") or "id" to set id = Precursor.Id and keep run separate.
unit	Character. Unit metadata stored in the returned object.
scope	Character. Scope metadata stored in the returned object.
q_source	Character. Label stored in metadata describing the q-value source.

Value

A dfdr_tbl with one row per run-by-precursor unit. The returned table includes id, run, is_decoy, q, pep, and score. Metadata are stored in attr(x, "meta").

Examples

```
library(tibble)

rep <- tibble(
  Run = c("r1", "r1", "r2", "r2"),
  Precursor.Id = c("P1", "P1", "P1", "P2"),
  Decoy = c(0L, 0L, 1L, 0L),
  Q.Value = c(0.01, 0.02, 0.03, 0.02),
  PEP = c(0.02, 0.01, 0.9, 0.05)
)

x <- diann_runxprecursor(rep, q_col = "Q.Value", run_col = "Run", id_mode = "runxid")
x
```

---

flag_headline *Flag diagnostic values based on heuristic thresholds*

Description

Adds simple, ASCII-only flags and a coarse overall status to a headline diagnostics table (typically produced by dfdr_headline or the headline table inside dfdr_run_all). The heuristics are intended for quick triage in reports; they do not replace manual review.

Usage

```
flag_headline(headline_tbl)
```

Arguments

headline_tbl	A data.frame/tibble of headline diagnostics. It may contain one or multiple rows. If some expected columns are missing, they are created and filled with NA.

Details

Rendering as icons (if desired) can be handled downstream (e.g. in an HTML template); this function returns plain text labels for portability.

Value

A tibble (or data frame) with the same rows as `headline_tbl` and additional columns:

flag_Dalpha Flag based on `D_alpha` (decoy support).

flag_CV Flag based on `CV_hat` (stability/variability).

flag_Dwin Flag based on `D_alpha_win` (local tail support).

flag_IPE Flag based on `IPE` (internal PEP calibration error).

flag_FDR Flag comparing `FDR_hat` to the nominal `alpha`.

flag_equalchance Flag based on `effect_abs` (equal-chance deviation).

status Overall triage status: `"VALID"`, `"CAUTION"`, or `"REVIEW_NEEDED"`.

interpretation A short human-readable interpretation string.

Examples

```
library(tibble)

headline_tbl <- tibble(
  list = c("A", "B"),
  alpha = 0.01,
  D_alpha = c(5, 80),
  CV_hat = c(0.35, 0.10),
  D_alpha_win = c(2, 30),
  IPE = c(0.12, 0.02),
  FDR_hat = c(0.02, 0.009),
  effect_abs = c(0.18, 0.05)
)

flag_headline(headline_tbl)
```

---

mokapot_competed_universe

Create a competed-winner universe from mokapot PSM outputs

Description

Constructs a "postprocessor universe" by selecting (competing) the single best scoring entry (target or decoy) per `run` + `spectrum_id` from mokapot PSM outputs. This yields one row per spectrum per run and can be used for downstream target-decoy diagnostics.

Usage

```
mokapot_competed_universe(
  raw,
  run_col = "run",
  spectrum_id_col = "spectrum_id",
  score_col = "mokapot score",
  q_col = "mokapot q-value",
  pep_col = "mokapot PEP",
  id_mode = c("runxid", "id"),
  unit = "psm",
  scope = "global",
  q_source = "mokapot q-value",
  q_max_export = 1
)
```

Arguments

raw	A combined mokapot table, typically the output of read_mokapot_psms.
run_col	Character. Column name for run (default "run").
spectrum_id_col	Character. Column name for spectrum identifier (default "spectrum_id").
score_col	Character. Column name for mokapot score used for competition (default "mokapot score"; higher is better).
q_col	Character. Column name for mokapot q-value (default "mokapot q-value").
pep_col	Character. Column name for mokapot PEP (default "mokapot PEP").
id_mode	Character. If "runxid" (default), sets id to "run||spectrum_id". If "id", sets id = spectrum_id.
unit	Character. Unit metadata stored in the returned object.
scope	Character. Scope metadata stored in the returned object.
q_source	Character. Source label stored in metadata.
q_max_export	Numeric. Export ceiling for q-values (typically 1 for mokapot).

Value

An object of class dfdr_tbl (a tibble subclass) with one row per competed run + spectrum_id. The returned table contains the standard columns required by diagFDR: id, is_decoy, q, pep, run, and score. Metadata are stored in attr(x, "meta").

Examples

```
library(tibble)

raw <- tibble(
  run = c("r1", "r1"),
  spectrum_id = c("1001", "1001"),
  `mokapot score` = c(2.0, 1.0),      # target wins
```

```
  `mokapot q-value` = c(0.01, 0.5),
  `mokapot PEP` = c(0.02, 0.6),
  is_target = c(TRUE, FALSE)
)

x <- mokapot_competed_universe(raw, id_mode = "runxid")
x
```

`print.dfdr_tbl` *Print a* `dfdr_tbl`

Description

Prints a compact header with attached metadata (unit, scope, q-source, etc.) and then falls back to the default tibble/data.frame print method.

Usage

```
## S3 method for class 'dfdr_tbl'
print(x, ...)
```

Arguments

`x`	A `dfdr_tbl`.
`...`	Passed to the next print method.

Value

Returns `x` invisibly (called for its printing side effect).

Examples

```
library(tibble)

df <- tibble(
  id = as.character(1:3),
  run = "run1",
  is_decoy = c(FALSE, TRUE, FALSE),
  score = c(10, 9, 8),
  q = c(0.01, 0.02, 0.03),
  pep = c(0.01, NA, 0.03)
)
x <- as_dfdr_tbl(df, unit = "psm", scope = "global", q_source = "toy_q")
x
```

`read_dfdr_maxquant_msms` *Read MaxQuant* `msms.txt` *into a* `dfdr_tbl` *(PSM-level; reconstructed TDC q-values)*

Description

Reads a MaxQuant `msms.txt` file and returns a `dfdr_tbl` at the PSM level. q-values are reconstructed by target-decoy counting (TDC) using MaxQuant `Score` (higher is better) and the `Reverse` column as the decoy indicator (`"+"` for decoy).

Usage

```
read_dfdr_maxquant_msms(
  path,
  pep_mode = c("drop", "sanitize", "strict"),
  exclude_contaminants = TRUE,
  add_decoy = 1L,
  unit = "psm",
  scope = "global",
  provenance = list()
)
```

Arguments

`path`	Character scalar. Path to MaxQuant `msms.txt`.
`pep_mode`	Character. How to handle the `PEP` column: `"drop"` Set `pep = NA` for all rows. `"sanitize"` Keep finite PEP in `[0,1]`; set other finite values to `NA` and warn. `"strict"` Error if any finite PEP is outside `[0,1]`.
`exclude_contaminants`	Logical. If `TRUE` and a `Potential contaminant` column exists, rows with `"+"` are removed. Default is `TRUE`.
`add_decoy`	Integer scalar. Additive correction in the TDC estimate: $\widehat{\mathrm{FDR}} = (D + add\_decoy)/T$. Default is 1.
`unit`	Character. Unit stored in the returned object metadata (default `"psm"`).
`scope`	Character. Scope stored in the returned object metadata (default `"global"`).
`provenance`	Named list. Stored in the returned object metadata.

Details

Required columns in `msms.txt`:

- `id` (unique PSM identifier; numeric or character)
- `Raw file` (run name)

- `Reverse` (MaxQuant decoy indicator; `"+"` for decoy; blank/NA for target)
- `Score` (MaxQuant score; higher is better)
- `PEP` (posterior error probability; optional in practice, see `pep_mode`)

The function computes q-values by target-decoy counting (TDC) using `Score` and the `Reverse` decoy label:

$$\widehat{\mathrm{FDR}}(i) = (D(i) + add\_decoy)/T(i), \qquad q(i) = \min_{j \geq i} \widehat{\mathrm{FDR}}(j)$$

Contaminants are not treated as decoys. If the column `Potential contaminant` exists and `exclude_contaminants = TRUE`, those rows are removed prior to computing q-values.

Value

A `dfdr_tbl` (tibble subclass) with one row per PSM and columns: `id`, `run`, `is_decoy`, `q`, `pep`, `score`. The q-values are reconstructed by TDC from `Score` and `Reverse`. Metadata are stored in `attr(x, "meta")` (including `unit`, `scope`, `q_source`, and `provenance`).

Examples

```
if (requireNamespace("readr", quietly = TRUE)) {
  # Create a tiny MaxQuant-like msms.txt and read it
  tmp <- tempfile(fileext = ".txt")
  txt <- paste0(
    "id\tRaw file\tReverse\tScore\tPEP\tPotential contaminant\n",
    "1\trun1\t\t100\t0.01\t\n",
    "2\trun1\t+\t90\t0.80\t\n",
   "3\trun1\t\t80\t0.02\t+\n",   # contaminant row removed when exclude_contaminants=TRUE
    "4\trun2\t\t70\t0.05\t\n",
    "5\trun2\t+\t60\t0.90\t\n"
  )
  writeLines(txt, tmp)

  x <- read_dfdr_maxquant_msms(tmp, pep_mode = "sanitize",
        exclude_contaminants = TRUE, add_decoy = 1L)
  x
  head(x$q)
}
```

`read_dfdr_mzid` *Read mzIdentML into a* `dfdr_tbl` *(generic; score-based TDC q-values)*

Description

Extracts a competed PSM universe (rank-1 by default) from an mzIdentML file, determines target/decoy labels, selects a single numeric PSM score CV term, and reconstructs q-values using target-decoy counting (TDC).

Usage

```
read_dfdr_mzid(
  mzid_path,
  rank = 1L,
  score_accession_preference = c("MS:1002257", "MS:1001330", "MS:1001328", "MS:1002052",
    "MS:1002049", "MS:1001331", "MS:1001171", "MS:1001950", "MS:1002466"),
  score_direction = c("auto", "lower_better", "higher_better"),
  add_decoy = 1L,
  min_score_coverage = 1,
  decoy_regex = "(^##|_REVERSED$|^REV_|^DECOY_)",
  unit = "psm",
  scope = NA_character_,
  provenance = list()
)
```

Arguments

`mzid_path`	Character scalar. Path to an mzIdentML file (`.mzid`).
`rank`	Integer scalar. Which `SpectrumIdentificationItem@rank` to use (default 1).
`score_accession_preference`	Character vector. Preferred PSI-MS CV accessions to treat as the primary PSM score, in priority order.
`score_direction`	One of `"auto"`, `"lower_better"`, `"higher_better"`. If `"auto"`, applies simple rules for common e-value/expectation accessions.
`add_decoy`	Integer scalar. Additive correction in the TDC FDR estimate: $\widehat{FDR} = (D + add\_decoy)/T$. Default 1.
`min_score_coverage`	Numeric in $(0, 1]$. Minimum fraction of PSMs required to have the chosen score CV term. Default 1.0 (strict; no missing scores allowed).
`decoy_regex`	Character scalar. Regex used to infer decoys from protein accessions if `PeptideEvidence@isDecoy` is not informative.
`unit`	Character. Stored in `dfdr_tbl` metadata (default `"psm"`).
`scope`	Character. Stored in `dfdr_tbl` metadata (default `NA`).
`provenance`	Named list. Stored in metadata (tool, version, parameters, command, etc.).

Details

This function is intended for workflows where mzIdentML does not provide explicit q-values or PEPs.

Value

A `dfdr_tbl` (tibble subclass) with one row per extracted PSM (at the requested `rank`). The returned object contains:

- `id`: a PSM identifier derived from run and spectrum ID (`"run||spectrumID"`);

- `is_decoy`: logical target/decoy label;
- `score`: an internal score where larger values mean better matches;
- `q`: reconstructed monotone q-values from TDC using `score` and `is_decoy`;
- `pep`: `NA` (mzIdentML PEPs are not parsed here);
- `run`: run identifier when available.

Metadata are stored in `attr(x, "meta")` (including `unit`, `scope`, `q_source`, and `provenance`).

Examples

```
if (requireNamespace("xml2", quietly = TRUE)) {
  # Minimal mzIdentML-like file sufficient for diagFDR's parser:
  # - 1 target and 1 decoy PSM at rank=1, with a numeric cvParam score
  tmp <- tempfile(fileext = ".mzid")
  mzid_txt <- paste0(
    "<?xml version='1.0' encoding='UTF-8'?>\n",
    "<MzIdentML xmlns='http://psidev.info/psi/pi/mzIdentML/1.1'>\n",
    "  <SequenceCollection>\n",
    "    <DBSequence id='DBSeq_t' accession='PROT1'/>\n",
    "    <DBSequence id='DBSeq_d' accession='REV_PROT2'/>\n",
    "    <PeptideEvidence id='PE_t' dBSequence_ref='DBSeq_t' isDecoy='false'/>\n",
    "    <PeptideEvidence id='PE_d' dBSequence_ref='DBSeq_d' isDecoy='true'/>\n",
    "  </SequenceCollection>\n",
    "  <DataCollection>\n",
    "    <AnalysisData>\n",
    "      <SpectrumIdentificationList>\n",
   "       <SpectrumIdentificationResult spectraData_ref='runA' spectrumID='scan=1'>\n",
    "          <SpectrumIdentificationItem rank='1'>\n",
    "            <PeptideEvidenceRef peptideEvidence_ref='PE_t'/>\n",
   "         <cvParam accession='MS:1001331' name='X!Tandem:hyperscore' value='50.0'/>\n",
    "          </SpectrumIdentificationItem>\n",
    "        </SpectrumIdentificationResult>\n",
   "       <SpectrumIdentificationResult spectraData_ref='runA' spectrumID='scan=2'>\n",
    "          <SpectrumIdentificationItem rank='1'>\n",
    "            <PeptideEvidenceRef peptideEvidence_ref='PE_d'/>\n",
   "         <cvParam accession='MS:1001331' name='X!Tandem:hyperscore' value='10.0'/>\n",
    "          </SpectrumIdentificationItem>\n",
    "        </SpectrumIdentificationResult>\n",
    "      </SpectrumIdentificationList>\n",
    "    </AnalysisData>\n",
    "  </DataCollection>\n",
    "</MzIdentML>\n"
  )
  writeLines(mzid_txt, tmp)

  x <- read_dfdr_mzid(tmp, rank = 1L, score_direction = "higher_better")
  x
  range(x$q)
}
```

read_diann_parquet *Read a DIA-NN* report.parquet

Description

Reads a DIA-NN `report.parquet` file using the **arrow** package (suggested dependency) and returns it as a tibble.

Usage

```
read_diann_parquet(path)
```

Arguments

`path`	Character scalar. Path to a DIA-NN `report.parquet` file.

Value

A tibble containing the columns present in the DIA-NN parquet report (column names depend on DIA-NN export settings).

Examples

```
if (requireNamespace("arrow", quietly = TRUE)) {
  # Create a tiny parquet file and read it back
  tmp <- tempfile(fileext = ".parquet")
  df <- data.frame(Precursor.Id = c("P1", "P2"), Q.Value = c(0.01, 0.02))
  arrow::write_parquet(df, tmp)
  out <- read_diann_parquet(tmp)
  out
}
```

read_mokapot_psms *Read mokapot PSM text outputs (targets + decoys) and combine*

Description

Reads the mokapot PSM output tables for targets and decoys (typically `*.mokapot.psms.txt` and `*.mokapot.decoy.psms.txt`) and concatenates them into a single table.

Usage

```
read_mokapot_psms(target_path, decoy_path)
```

Arguments

target_path	Character scalar. Path to the mokapot target PSM table (e.g. *.mokapot.psms.txt).
decoy_path	Character scalar. Path to the mokapot decoy PSM table (e.g. *.mokapot.decoy.psms.txt).

Value

A tibble containing all rows from the target and decoy tables. Column names and content are determined by mokapot.

Examples

```
if (requireNamespace("readr", quietly = TRUE)) {
  # Create tiny example mokapot-like target/decoy tables and read them back
  tf1 <- tempfile(fileext = ".txt")
  tf2 <- tempfile(fileext = ".txt")
  txt1 <- "run\tspectrum_id\tmokapot score\tmokapot q-value\tmokapot PEP
  \tis_target\nr1\t1\t2.0\t0.01\t0.02\tTRUE\n"
  txt2 <- "run\tspectrum_id\tmokapot score\tmokapot q-value\tmokapot PEP
  \tis_target\nr1\t1\t1.0\t0.50\t0.60\tFALSE\n"
  writeLines(txt1, tf1)
  writeLines(txt2, tf2)

  raw <- read_mokapot_psms(tf1, tf2)
  raw
}
```

---

read_spectronaut *Read a Spectronaut TSV report*

Description

Reads a Spectronaut report exported as tab-separated values (TSV). Uses data.table::fread() for efficient reading of large files.

Usage

```
read_spectronaut(path, dec = ".", select = NULL)
```

Arguments

path	Character scalar. Path to a Spectronaut report TSV file.
dec	Character scalar. Decimal separator passed to fread() (default ".").
select	Optional character vector of column names to read; NULL reads all columns.

Details

Spectronaut may use a comma as decimal separator in some locales; set dec="," when needed.

Value

A tibble containing the columns present in the Spectronaut report (column names depend on the export configuration).

Examples

```
if (requireNamespace("data.table", quietly = TRUE)) {
  # Create a tiny TSV and read it back
  tmp <- tempfile(fileext = ".tsv")
  txt <- paste0(
    "R.FileName\tEG.PrecursorId\tEG.IsDecoy\tEG.Qvalue\tEG.PEP\tEG.Cscore\n",
    "run1\tP1\tFalse\t0.01\t0.02\t120\n",
    "run1\tP2\tTrue\tNaN\t0.90\t10\n"
  )
  writeLines(txt, tmp)
  read_spectronaut(tmp)
}
```

---

read_spectronaut_efficient

Read Spectronaut efficiently with column selection

Description

Convenience wrapper around `read_spectronaut`. For very large Spectronaut reports, reading only a minimal set of columns can substantially speed up I/O and reduce memory usage.

Usage

```
read_spectronaut_efficient(path, minimal = TRUE, dec = ".")
```

Arguments

`path`	Character scalar. Path to a Spectronaut TSV report.
`minimal`	Logical. If `TRUE` (default), reads only columns typically needed for precursor-level diagnostics; if `FALSE`, reads all columns.
`dec`	Character scalar. Decimal separator (use `","` for some locales).

Value

A tibble containing either the selected minimal columns (if `minimal=TRUE`) or all columns (if `minimal=FALSE`).

Examples

```
if (requireNamespace("data.table", quietly = TRUE)) {
  tmp <- tempfile(fileext = ".tsv")
  txt <- paste0(
   "R.FileName\tR.Condition\tEG.PrecursorId\tEG.IsDecoy\tEG.Qvalue\tEG.PEP\tEG.Cscore\n",
    "run1\tcondA\tP1\tFalse\t0.01\t0.02\t120\n",
    "run1\tcondA\tP2\tTrue\tNaN\t0.90\t10\n"
  )
  writeLines(txt, tmp)
  read_spectronaut_efficient(tmp, minimal = TRUE)
}
```

`score_to_pvalue` *Convert identification scores to p-values or pseudo-p-values*

Description

Converts identification scores to:

- **p-values** when the score has a known p-value definition (e.g. Mascot score),
- **pseudo-p-values** when no such definition exists (e.g. MaxQuant/Andromeda scores).

Usage

```
score_to_pvalue(
  score,
  method = c("mascot", "neglog10p", "evalue", "rank", "decoy_ecdf"),
  is_decoy = NULL,
  eps = .Machine$double.xmin,
  clamp = TRUE,
  ties = c("average", "first", "random", "max", "min")
)
```

Arguments

`score`	Numeric vector. Identification scores. Higher-is-better is assumed for `method = "rank"` and `method = "decoy_ecdf"` (flip sign upstream if needed).
`method`	Character. Conversion method:
	`"mascot"` Treat `score` as Mascot score $S = -10\log_{10}(p)$; returns $p = 10^{-S/10}$.
	`"neglog10p"` Generic $S = -10\log_{10}(p)$; same transform as `"mascot"`.
	`"evalue"` Treat `score` as an expectation/e-value-like quantity and return it as a p-like value.
	`"rank"` Rank-based pseudo-p-values $p_i = \mathrm{rank}(-s_i)/(n+1)$.
	`"decoy_ecdf"` Decoy-ECDF pseudo-p-values from the empirical right-tail of decoy scores; requires `is_decoy`.

is_decoy	Optional logical vector (same length as score). Required for method = "decoy_ecdf". TRUE=decoy, FALSE=target.
eps	Numeric scalar > 0. Lower bound to avoid exact 0 (default .Machine$double.xmin).
clamp	Logical. If TRUE (default), clamp output to [0,1].
ties	Character. Tie-handling for method="rank"; passed to rank().

Details

Pseudo-p-values are useful as inputs to plausibility checks (e.g. calibration/ECDF diagnostics), but they are not guaranteed to be valid null p-values unless the method is explicitly null-based (e.g. method = "decoy_ecdf" with a reliable decoy set).

Value

A numeric vector of p-values/pseudo-p-values of the same length as score. Non-finite scores yield NA_real_. For method="decoy_ecdf", returned values estimate the decoy right-tail probability $P(S_{decoy} \geq s)$ (with a small +1 smoothing), which can be interpreted as a null-calibrated pseudo-p-value when decoys provide a good null sample.

Examples

```
set.seed(1)
score <- rnorm(10)

# Mascot-like transformation: S = -10 log10(p)
S <- c(10, 20, 30)
score_to_pvalue(S, method = "mascot")

# Rank-based pseudo-p-values (higher score => smaller p)
score_to_pvalue(score, method = "rank")

# Decoy-ECDF pseudo-p-values (requires decoys)
n <- 200
score2 <- c(rnorm(n, mean = 0), rnorm(n, mean = 1))  # targets shifted higher
is_decoy <- c(rep(TRUE, n), rep(FALSE, n))
p2 <- score_to_pvalue(score2, method = "decoy_ecdf", is_decoy = is_decoy)
summary(p2)
```

---

spectronaut_runxprecursor

Spectronaut -> run-by-precursor universe (deduplicated within run by EG.PrecursorId)

Description

Constructs a run-wise precursor universe from a Spectronaut report. The output is suitable for run-wise FDR diagnostics because each row corresponds to a run × precursor hypothesis.

Usage

```
spectronaut_runxprecursor(
  rep,
  q_col = "EG.Qvalue",
  run_col = NULL,
  pep_col = "EG.PEP",
  score_col = "EG.Cscore",
  q_max_export = 1,
  recompute_q = TRUE,
  id_mode = c("id", "runxid"),
  unit = "runxprecursor",
  scope = "runwise",
  q_source = q_col
)
```

Arguments

`rep`	Spectronaut report tibble (e.g. returned by `read_spectronaut` or `read_spectronaut_efficient`).
`q_col`	Character. q-value column name (default `"EG.Qvalue"` for run-wise control).
`run_col`	Optional character. Run column name; if `NULL`, attempts to detect one of `"R.FileName"` or `"R.Condition"`.
`pep_col`	Optional character. PEP column name (default `"EG.PEP"`).
`score_col`	Optional character. Score column name (default `"EG.Cscore"`).
`q_max_export`	Numeric. Export ceiling for q-values (default 1.0), stored in metadata.
`recompute_q`	Logical. If `TRUE` (default) and decoys lack finite q-values, recompute q-values from `score_col` per run.
`id_mode`	Character. `"id"` (default) uses `id = EG.PrecursorId` and keeps `run` separate; `"runxid"` uses `id = "run||precursor"` for global uniqueness.
`unit`	Character. Unit metadata stored in the returned object.
`scope`	Character. Scope metadata stored in the returned object.
`q_source`	Character. Source label stored in metadata.

Details

If Spectronaut exports `NaN` q-values for decoys (common), the function can optionally recompute q-values from scores per run using target-decoy competition.

When Spectronaut exports decoys with `NaN` q-values, recomputing q-values from scores is performed *per run* to respect the run-wise hypothesis structure and to avoid pooling scores across runs that may not be comparable.

Value

A `dfdr_tbl` (tibble subclass) with one row per `run` × `precursor`. The returned table contains the standard columns required by `diagFDR`: `id`, `is_decoy`, `q`, `pep`, `run`, `score`. Metadata (including whether q-values were recomputed) are stored in `attr(x, "meta")`.

Examples

```
library(tibble)

# Minimal toy Spectronaut-like report
rep <- tibble(
  R.FileName = c("run1","run1","run1","run2","run2","run2"),
  EG.PrecursorId = c("P1","P2","P3","P1","P2","P4"),
  EG.IsDecoy = c("False","True","False","False","True","False"),
  EG.Qvalue = c(0.01, NaN, 0.02, 0.03, NaN, 0.04),
  EG.PEP = c(0.02, 0.9, 0.05, 0.06, 0.95, 0.08),
  EG.Cscore = c(120, 10, 80, 100, 5, 60)
)

x <- spectronaut_runxprecursor(
  rep,
  q_col = "EG.Qvalue",
  run_col = "R.FileName",
  recompute_q = TRUE,
  id_mode = "runxid"
)
x
```

Index